Replaneamiento en Agentes Inteligentes. Revisión de Grafos de Planning.

Gerardo A. PARRA

Departamento de Informática y Estadística
UNIVERSIDAD NACIONAL DEL COMAHUE
e-mail: gparra@uncoma.edu.ar

Guillermo R. SIMARI

Departamento de Ciencias de la Computación
UNIVERSIDAD NACIONAL DEL SUR
e-mail:grs@cs.uns.edu.ar

Palabras Clave: INTELIGENCIA ARTIFICIAL, PLANEAMIENTO, DINÁMICA DE CREENCIAS

Resumen

Los agentes inteligentes autónomos, por su proactividad, se ven obligados a considerar la satisfacción de sus metas a través de un conjunto estructurado de acciones que conforman un plan. El modelo BDI (*Belief, Desires and Intentions*) para representar el conjunto cognitivo de un agente es una posibilidad interesante que permite estudiar el problema que introduce el dinamismo natural del entorno en el que un plan particular se desenvuelve.

El dinamismo del entorno provoca que algunos de los planes deban ser modificados para poder alcanzar las metas finales. Es esta una actividad de replaneamiento.

Esta propuesta de trabajo postula la conveniencia de adoptar el punto de vista del área de Dinámica de Creencias al considerar la actividad de replaneamiento de un agente inteligente. En trabajos previos se ha introducido un modelo para representar expansiones y contracciones de grafos de planning. En este trabajo retomamos aquel modelo y nos proponemos representar situaciones en las cuales es necesario remover algunas piezas del plan global y reemplazarlas por subplanes convenientes que ofrezcan la posibilidad de éxito para el plan global. Tales situaciones serán consideradas operaciones de revisión de grafos de planning.

1 Introducción

Los agentes inteligentes autónomos, por su proactividad, se ven obligados a considerar la satisfacción de sus metas a través de un conjunto estructurado de acciones que conforman un plan. El modelo BDI (*Belief, Desires and Intentions*)[10] para representar el conjunto cognitivo de un agente es una posibilidad interesante que permite estudiar el problema que introduce el dinamismo natural del entorno en el que un plan particular se desenvuelve. El entorno corriente, el mundo actual del agente, es representado con un modelo de creencias (beliefs) adecuado. Las metas del agente representan sus deseos (desires) y describen en forma parcial estados del entorno preferidos. Finalmente, los planes para alcanzar alguno de aquellos estados constituyen, en cierta forma, las intenciones.

El dinamismo del entorno provoca que algunos de los planes deban ser modificados para poder alcanzar las metas finales. Ciertas partes pueden ser conservadas, pero otras deben ser removidas y reemplazadas por subplanes convenientes que ofrezcan la posibilidad de éxito para el plan global.

Esta propuesta de trabajo postula la conveniencia de adoptar el punto de vista del área de Dinámica de Creencias[5,9] al considerar la actividad de replaneamiento de un agente inteligente. En [11] se estableció un modelo para representar expansiones en grafos de planning. Este fenómeno tiene lugar cuando es necesario incorporar nuevas acciones para poder cumplir las metas. La característica fundamental de la operación de expansión es que permite reutilizar, en gran medida, el grafo de planning original. En [12] se ha definido un operador de contracción de grafos de planning con el objetivo de representar situaciones en las cuales es necesario remover algunas piezas del plan global debido, por ejemplo, a la imposibilidad de llevarlas a cabo. En este trabajo, se introduce un operador que modela la revisión de grafos de planning. Este fenómeno se produce cuando es necesario remover algunas piezas del plan y reemplazarlas por elementos adecuados que ofrezcan la posibilidad de éxito para el plan general.

El trabajo está organizado de la siguiente manera. En la sección 2 se presentan los principales conceptos relacionados al área de planning en Inteligencia Artificial. Se analiza un dispositivo de planning particular y se trata, en este contexto, la construcción de grafos de planning. En la siguiente sección presentamos los aspectos más relevantes relacionados con la dinámica de creencias. La sección 4 contiene las principales contribuciones de este trabajo. Se define un operador de revisión de grafos de planning, se introducen sus propiedades deseables y su definición constructiva. Finalmente, la sección 5 incluye las conclusiones del trabajo, así como también las consideraciones sobre trabajos futuros.

2 Un Dispositivo de Planning: Graphplan

El objetivo central del área de *planning* en el contexto de Inteligencia Artificial es construir algoritmos que hagan posible a un agente elaborar un curso de acción para lograr sus metas. El resultado producido por un dispositivo de planning (*planner*) es una secuencia de acciones las cuales, cuando son ejecutadas en un mundo que satisface la descripción del estado inicial, lograrán la obtención de la meta. Existe una amplia variedad de lenguajes para representar el mundo, las metas del agente y las acciones posibles. En este trabajo adoptamos, en primera instancia, la representación STRIPS[1] como lenguaje de representación.

La representación STRIPS describe el estado inicial del mundo mediante un conjunto completo de literales básicos (*ground*) y las metas son definidas como una conjunción proposicional. La teoría de dominio, es decir, la descripción formal de las acciones disponibles para el agente, completa la descripción del problema de planning.

En la representación STRIPS, cada acción es descripta por dos fórmulas: la *precondición* y el *efecto* o *poscondición*. Ambas están constituídas por una conjunción de literales y definen una función de transición de un mundo a otro. Una acción puede ser ejecutada en cualquier mundo *w* que satisfaga la fórmula de precondición. El resultado de ejecutar una acción en un mundo *w* es especificado tomando la descripción de *w*, adicionando cada literal de la poscondición de la acción y eliminando literales contradictorios.

Lo expuesto hasta aquí caracteriza un problema de planning clásico. Es importante destacar que existen muchas suposiciones que simplifican el problema: tiempo atómico, no existen eventos exógenos, los efectos de las acciones son determinísticos, omnisciencia por parte del agente, etc..

Graphplan[2, 3] es un simple y elegante algoritmo que produce un planner extremadamente eficiente. En muchos casos, órdenes de magnitud más eficiente que sistemas previos.

El funcionamiento de Graphplan alterna entre dos fases: la *construcción del grafo de planning* y la *extracción de la solución*. La primera fase construye, a través de sucesivas etapas, un *grafo de planning* hacia delante en el tiempo hasta que se logra una condición necesaria (pero que puede ser insuficiente) para la existencia de un plan. Luego, la fase de extracción de solución realiza un recorrido hacia atrás sobre el grafo, buscando un plan que resuelva el problema. Si no es hallada una solución, el ciclo se repite llevando a cabo una nueva etapa en la construcción del grafo de planning.

2.1 Grafos de Planning

El grafo de planning contiene dos tipos de nodos: nodos de proposición y nodos de acción, organizados en niveles. Los niveles con numeración par contienen nodos de proposición (es decir, literales ground) y, en particular, el nivel cero consiste precisamente de las proposiciones que son verdaderas en el estado inicial del problema de planning. Los nodos presentes en niveles con numeración impar corresponden a instancias de acción. Existe uno de tales nodos por cada instancia de acción cuyas precondiciones estén presentes (y sean mutuamente consistentes) en el nivel previo.

Los nodos correspondientes a instancias de acción están conectados mediante arcos a los nodos de proposición (en el nivel anterior) que constituyen las precondiciones de la acción. Existen arcos adicionales (arcos de poscondición) que conectan los nodos de acción con los nodos de proposiciones (en el nivel siguiente) que se hacen verdaderas por efecto de la acción.

Es importante destacar que el grafo de planning representa acciones que pueden desarrollarse ‘*en paralelo*’ en cada nivel de acciones. Sin embargo, el hecho que dos acciones estén presentes en el mismo nivel de un grafo de planning *no* significa que sea posible ejecutar ambas a la vez.

Central a la eficiencia de Graphplan es la inferencia de una relación binaria de exclusión mutua, denominada ‘*mutex*’, entre nodos presentes en el mismo nivel. La relación se define recursivamente como sigue:

- Dos instancias de acción en el nivel *i* son *mutex* si ocurre alguno de los casos siguientes:

    - el efecto de una acción es la negación del efecto de la otra (*Efectos Inconsistentes*).
    - el efecto de una acción elimina la precondición de la otra (*Interferencia*).
    - las acciones tienen precondiciones que son mutuamente exclusivas a nivel *i – 1* (*Necesidades Conflictivas*).

- Dos proposiciones en el nivel *i* son *mutex* si una es la negación de la otra o si todas las formas de arribar a estas proposiciones (es decir, las acciones en el nivel *i – 1*) son *mutex* tomadas de a dos (S*oporte Inconsistente*).

A continuación presentamos, a modo de ejemplo, la especificación STRIPS del *dinner-date problem*[4].

```
Condiciones Iniciales:  (and (basura) (manosLimpias) (silencio))
Meta:  (and (cena) (regalo) (not (basura)))
Acciones:

  Cocinar                 :precondition (manosLimpias)
                          :effect (cena)
  Envolver                :precondition (silencio)
                          :effect (regalo)
  Llevar_a_mano           :precondition
                          :effect   and ( (not (basura)) (not (manosLimpias)))
  Llevar_en_carretilla    :precondition
                          :effect   and ( (not (basura)) (not (silencio)))
```

Figura 1

El *dinner-date problem* consiste en encontrar un plan para preparar una cita sorpresa para nuestra amada que se encuentra durmiendo. La meta del problema es sacar las bolsas de basura, preparar la cena y envolver el regalo. Existen cuatro acciones posibles: cocinar, envolver, llevar_a_mano y llevar_en_carretilla. Cocinar requiere manosLimpias y produce cena. Envolver tiene como precondición silencio porque es una sorpresa y produce regalo. Llevar_a_mano elimina basura pero, el contacto manual con las bolsas, niega manosLimpias. La acción final, llevar_en_carretilla, también elimina basura pero, a causa del desplazamiento ruidoso, niega silencio. Inicialmente, tenemos manosLimpias, la casa tiene basura y está en silencio. Las demás proposiciones se consideran falsas.

El grafo de planning se construye de la siguiente manera. Todas las condiciones iniciales son ubicadas en el primer nivel de proposiciones (nivel cero) del grafo. Construir un nivel de acción genérico consiste en lo siguiente. Para cada operador y para cada forma de instanciar las precondiciones de ese operador a proposiciones del nivel previo, se inserta un nodo de acción si no existen dos precondiciones que sean mutuamente exclusivas. Además, se insertan todas las acciones de mantenimiento (acciones nulas) y los arcos de las precondiciones. Luego se chequea la relación de exclusión mutua entre los nodos de acción y se crea una lista, que mantiene esas relaciones, para cada acción. Para crear un nivel de proposiciones genérico, simplemente se tienen en cuenta todos los efectos de las acciones presentes en el nivel previo (incluyendo las acciones de mantenimiento) y se los ubica en el siguiente nivel como proposiciones, conectándolos vía los apropiados arcos de

poscondición. Finalmente, dos proposiciones son marcadas como mutuamente exclusivas, si todas las formas de generar la primera son mutuamente exclusivas con respecto a todas las formas de generar la segunda de las proposiciones marcadas.

A continuación se exhibe el grafo de planning para el *dinner-date problem* expandido un nivel de acción y un nivel de proposición, a partir del nivel cero.

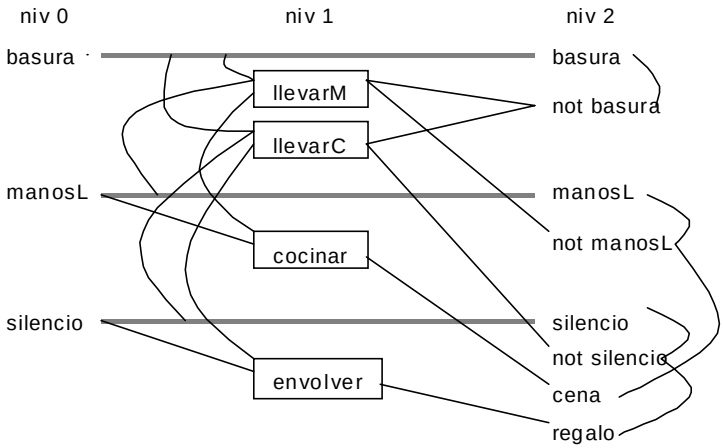

Figura 2

En el gráfico anterior, los nombres de las acciones están encerrados por un cuadro y las líneas horizontales grises entre diferentes niveles de proposiciones representan acciones de mantenimiento que modelan persistencia. Las líneas curvas entre acciones y proposiciones en un nivel particular denotan relaciones de exclusión mutua (*mutex*). Notemos que la acción llevar_a_mano es mutex con la persistencia de basura porque tienen efectos inconsistentes. Llevar_en_carretilla es mutex con envolver por interferencia, dado que la primera acción elimina silencio. En el segundo nivel de proposiciones, not silencio es mutex con regalo a causa de soporte inconsistente. Recordemos que la meta de este problema es lograr la conjunción de not basura, cena y regalo. Dado que la totalidad de estos literales están presentes en el segundo nivel proposicional y, dado que ninguno de ellos es mutex con cada uno de los otros, existe la posibilidad de la existencia de un plan para resolver el problema. En este caso, la segunda fase de Graphplan es ejecutada: la extracción de la solución.

2.2 Extracción de la Solución

La fase de extracción de la solución se aboca a la tarea de hallar un plan considerando cada una de las *n* submetas -términos de la conjunción- que conforman la meta. Para cada uno de tales literales presentes en un nivel *i*, Graphplan elige una acción *a* en el nivel *i – 1* tal que produzca esa submeta. Este es un punto de *backtracking*: si más de una acción produce una submeta, entonces Graphplan debe considerar a todas ellas con el fin de asegurar completitud. Si *a* es consistente (es decir, no mutex) con todas las acciones que han sido escogidas hasta ahora en este nivel, entonces se procede a la siguiente submeta. De lo contrario, se realiza *backtracking* a la elección previa.

Una vez que ha encontrado un conjunto consistente de acciones en el nivel *i – 1*, Graphplan trata recursivamente de encontrar un plan para el conjunto formado a partir de la unión de todas las

precondiciones de las aquellas acciones en el nivel *i – 2*. El caso base de la recursión es el nivel cero: si las proposiciones están presentes en ese nivel entonces Graphplan ha encontrado una solución. De lo contrario, si el backtracking falla para todas las combinaciones de las posibles acciones que soportan cada submeta (para cada nivel), entonces Graphplan extiende el grafo de planning con un nuevo nivel de acción y un nuevo nivel de proposición y luego intenta nuevamente la extracción de una solución.

En el ejemplo planteado, existen tres submetas en el nivel dos. Not basura es soportada por llevar_a_mano y por llevar_en_carretilla; cena es soportada por cocinar y regalo es soportada por envolver. De esta manera, Graphplan debe considerar dos conjuntos de acciones en el nivel uno: {llevar_a_mano,cocinar,envolver} y {llevar_en_carretilla,cocinar,envolver}. Pero, desafortunadamente, ninguno de estos conjuntos es consistente porque llevar_a_mano es mutex con cocinar y llevar_en_carretilla es mutex con envolver. Por lo tanto, la extracción de la solución falla y Graphplan extiende el grafo de planning hasta el nivel cuatro como vemos a continuación.

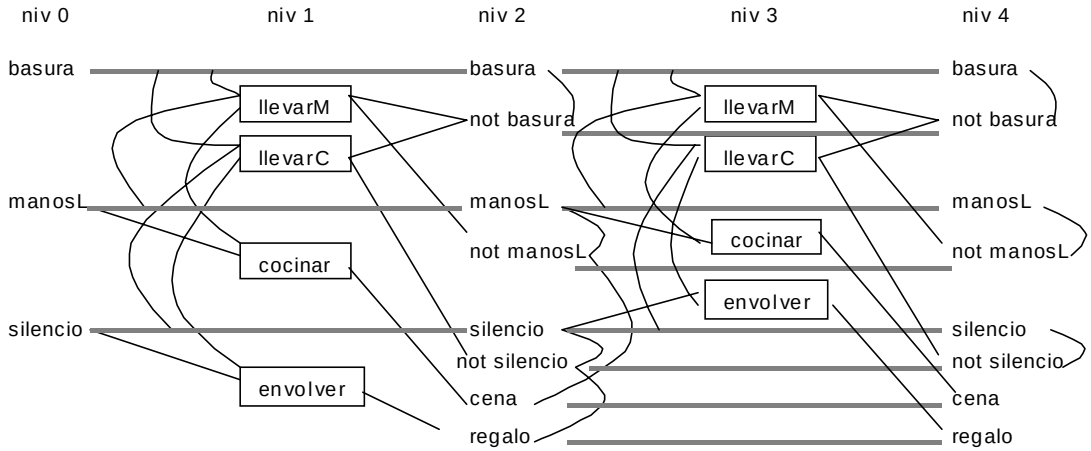

Figura 3

Note la diferencia entre los niveles dos y cuatro del grafo de planning. Aunque no aparecen nuevos literales en el cuarto nivel, existen menos relaciones de exclusión mutua. Por ejemplo, cena y not manosLimpias ya no son mutuamente exclusivas. La diferencia más importante se da a nivel tres: existen cinco acciones de mantenimiento adicionales que modelan la posible persistencia de los literales logrados en el nivel dos. Esto significa que cada una de las submetas tiene acciones de soporte adicionales a considerar durante el proceso de extracción de la solución. Específicamente,

- Not basura es soportada por llevar_a_mano, llevar_en_carretilla y por una acción de mantenimiento.
- Cena es soportada por cocinar y por una acción de mantenimiento.
- Regalo es soportada por envolver y por una acción de mantenimiento.

Este incremento de la flexibilidad permite a la fase de extracción de solución encontrar un plan. Existen varias combinaciones que funcionan, ilustramos una a continuación.

Elegimos como soporte de not basura a llevar_a_mano, soportamos cena con la acción de mantenimiento y se soporta regalo con envolver. Ninguna de estas acciones es mutex con otra, por lo tanto las acciones seleccionadas para el nivel tres son consistentes. La selección de estas acciones

nos conduce a las siguientes submetas para el nivel dos: cena (precondición de la relación de mantenimiento) y silencio (precondición de envolver); dado que llevar_a_mano no tiene precondiciones, existen sólo dos submetas de nivel dos. A raíz del proceso recursivo de extracción de solución, se selecciona cocinar para soportar cena y la relación de mantenimiento para soportar silencio; estas dos acciones no son mutex, por lo tanto las elecciones para el nivel uno son consistentes. Las precondiciones de estas acciones crean dos submetas para el nivel cero: manosLimpias y silencio. Dado que estas proposiciones están presentes en las condiciones iniciales, la selección es consistente y existe un plan para solucionar el problema.

Es importante destacar que Graphplan genera un plan inherentemente paralelo o parcialmente ordenado. Las acciones seleccionadas para el nivel tres, llevar_a_mano y envolver pueden ser ejecutadas en cualquier orden y producirán el mismo efecto. Así, si uno desea una secuencia de acciones totalmente ordenada como plan definitivo, uno podría elegir arbitrariamente: cocinar, llevar_a_mano y envolver.

Retornemos, por un instante, al *dinner-date problem* (Figura 1). ¿Qué sucedería si, una vez construído y ejecutado el plan definitivo, el agente descubre que alguna de las acciones necesarias para lograr la meta no produjo los resultados esperados? A modo de ejemplo, supongamos que la acción cocinar no ha sido ejecutada adecuadamente y, como consecuencia de ello, se ha *quemado la cena*. Ante esta situación sería necesaria una tarea de replaneamiento. El mayor problema, desde un punto de vista computacional, consiste en que el grafo de planning debe volver a construirse desde cero para intentar encontrar nuevamente la solución. En las secciones subsiguientes, consideraremos la conveniencia de adoptar el punto de vista del área de Dinámica de Creencias con el fin de intentar simplificar este problema.

3 Dinámica de Creencias

Dinámica de creencias es el proceso por el cual un agente cambia sus creencias, realizando una transición desde un estado epistémico a otro. Cuando tal agente aprende nueva información puede concluir que esta información contradice sus creencias previas. En este caso, el agente debe revisar sus creencias y decidir cuáles de sus creencias previas tienen que ser abandonadas con el fin de incorporar la nueva información.

Uno de las más fundamentales aproximaciones a la formalización de la dinámica de creencias es el modelo AGM[5]. En este enfoque, los estados epistémicos son representados por conjuntos de creencias que son conjuntos de sentencias cerrados bajo consecuencia lógica.

Notación: Se adopta un lenguaje proposicional L con un conjunto completo de conectivos booleanos: negación, conjunción, disyunción e implicación. Las fórmulas en L serán denotadas por letras griegas minúsculas y los conjuntos de sentencias en L serán denotadas mediante letras mayúsculas. Se emplea un operador de consecuencia Cn. Cn toma un conjunto de sentencias en L y produce un nuevo conjunto de sentencias. Se asume que el operador Cn satisface las propiedades de *inclusión* ($A \subseteq Cn(A)$), *iteración* ($Cn(A) = Cn(Cn(A))$) y monotonicidad (si $A \subseteq B$ entonces $Cn(A) \subseteq Cn(B)$).

Sea $\mathbf{K} = Cn(\mathbf{K})$ un conjunto de creencias y α una sentencia en un lenguaje proposicional L. Los tres principales tipos de operaciones de cambio de creencias son los siguientes[6]:

- **Expansión**: Una nueva sentencia es incorporada a un estado epistémico. Si '+' es un operador de expansión entonces **K** + α denota el conjunto de creencias **K** expandido por α.
- **Contracción**: Alguna sentencia presente en el estado epistémico es retraída sin incorporar nueva información. Si '-' es un operador de contracción entonces **K** - α denota el conjunto de creencias **K** contraído por α.
- **Revisión**: Una nueva sentencia es incorporada de manera consistente al estado epistémico. Con el fin de hacer posible esta operación, algunas sentencias deben ser retraídas del estado epistémico original. Si '*' es un operador de revisión entonces **K** * α denota el conjunto de creencias **K** revisado por α.

Gärdenfors[6] propone los siguientes postulados básicos de racionalidad para la operación de revisión:

(**K*** 1) **Clausura**: **K** * α = *Cn*(**K** * α).

(**K*** 2) **Exito**: $\alpha \in$ (**K** * α).

(**K*** 3) **Inclusión**: **K** * $\alpha \subseteq$ **K** + α.

(**K*** 4) **Vacuidad**: Si **K** no deriva $\neg\alpha$ entonces **K** * α = **K** + α.

(**K*** 5) **Consistencia**: Si $\neg\alpha$ no es un teorema entonces **K** * α no es inconsistente.

(**K*** 6) **Extensionalidad**: Si α y β son lógicamente equivalentes entonces **K** * α = **K** * β.

La operación de expansión puede ser definida explícitamente tomando la clausura lógica del conjunto de creencias **K** unido a la sentencia α. Es decir,

$$(\mathbf{K} + \alpha) = Cn(\mathbf{K} \cup \{\alpha\}).$$

Las operaciones de contracción y revisión pueden ser definidas usando nociones lógicas y algún mecanismo de selección. Además, mediante las identidades de *Levi* y de *Harper*[6], es posible definir una operación en función de la restante.

No desarrollaremos los postulados de racionalidad propuestos para las operaciones de expansión y contracción, puesto que no son necesarios para el resto del paper. El lector interesado, puede consultar las referencias citadas en esta sección. Además, una presentación exhaustiva de los diferentes modelos de cambios de creencias puede encontrarse en [7,8].

4 Revisión de Grafos de Planning

Asumamos que un agente descubre que, una de las acciones requeridas en el plan definitivo no pudo ser ejecutada adecuadamente, i.e. no ha dado los resultados esperados. Ante esta situación, una porción del plan debe ser removida y reemplazada por un subplan conveniente que ofrezca la posibilidad de éxito para el plan global.

Analicemos un ejemplo concreto tomando como base el *dinner-date problem* (Figura 1). Supongamos que, una vez construído y ejecutado el plan definitivo, el agente descubre que se ha *quemado la cena*. Ante tal situación, una posibilidad interesante sería apelar a una nueva acción que produzca el efecto esperado. Supongamos que se dispone, para tal fin, de una acción llamar_rotiseria con poscondición cena y sin precondiciones. De esta manera, reemplazar la porción del plan que no produjo los resultados esperados por un subplan alternativo ofrece la posibilidad de éxito al plan general. Sin embargo, esto implica, en el contexto de Graphplan, la reconstrucción del grafo de planning desde el nivel cero.

Como vimos en las secciones previas, la construcción del grafo de planning para un problema determinado no es una tarea trivial. Por lo tanto, sería interesante conservar buena parte del grafo ante una modificación del problema original.

Con esta motivación, proponemos en este trabajo la definición de una operación de revisión para grafos de planning. En primer lugar, necesitamos tratar a cada acción y a todas sus precondiciones y poscondiciones como una unidad.

Definición 4.1. Un *esquema de acción* es una terna (*Pre*,*a*,*Pos*) donde, *a* es una acción, *Pre* es un conjunto finito de proposiciones que constituyen las precondiciones de *a* y *Pos* es un conjunto finito de proposiciones que se verifican como resultado de aplicar la acción *a* (poscondiciones de *a*).

Notación: Sea *A* un esquema de acción. Emplearemos A_{Pre} y A_{Pos} para denotar, respectivamente, las precondiciones y poscondiciones de *A*.

A continuación, necesitamos especificar cuándo un esquema de acción pertenece a un nivel determinado de un grafo de planning.

Definición 4.2. Sea Π un grafo de planning. Sea *n* un nodo de proposición o de acción perteneciente al grafo. Mediante la función $Lev_{\Pi}(n)$ se indica el nivel correspondiente al nodo *n* en el grafo Π. Además, $Lev_{\Pi}(n)$ es indefinido si y sólo si el nodo *n* no pertenece a Π.

Definición 4.3. Sea Π un grafo de planning y sea *A* = (*Pre*,*a*,*Pos*) un esquema de acción. Decimos que *A pertenece a nivel i* a Π si y sólo si $Lev_{\Pi}(a) = i$, cada uno de los elementos de los conjuntos *Pre* y *Pos* existen como nodos en el grafo y si existen los arcos que modelan las relaciones correspondientes.

En este punto, ya estamos en condiciones de definir un operador de revisión de grafos de planning.

Definición 4.4. Sea $\boldsymbol{\Pi}$ la clase de los grafos de planning y sea *A* = (*Pre*,*a*,*Pos*) un esquema de acción. El *operador de revisión a nivel i*, denotado $\otimes^i$, se define de la siguiente manera:

$$\otimes^i : \boldsymbol{\Pi} \times A \rightarrow \boldsymbol{\Pi}.$$

La tarea básica del operador $\otimes^i$ es obtener, dado un grafo de planning y un esquema de acción *A*, un nuevo grafo de planning Π'. Este nuevo grafo tiene dos características principales: el esquema *A* pertenece a nivel *i* a Π' y, posiblemente, esquemas de acción adicionales han sido removidos de Π'.

La operación de revisión de grafos de planning debería entenderse como un proceso mediante el cual se reemplaza un esquema de acción *B* por un esquema *A* que contiene las poscondiciones de *B*. Sin embargo, es posible que el esquema a reemplazar y el nuevo esquema difieran en sus precondiciones.

4.1 Postulados del Operador de Revisión

Sea Π un grafo de planning y sean *A* y *B* esquemas de acción. A continuación se presentan las propiedades deseables de la revisión a nivel *i* de Π por *A*.

(**R 1**) **Exito**: *A* pertenece a nivel *i* a $(\Pi \otimes^i A)$.

El postulado de éxito es una consecuencia del comportamiento esperado del operador de revisión y del postulado de éxito para expansiones de grafos de planning[11].

(**R 2**) **Inclusión**: $(\Pi \otimes^i A) \subseteq (\Pi \oplus^i A)$.

Este postulado se basa en que la expansión simplemente inserta un nuevo esquema de acción al grafo de planning mientras que la revisión puede requerir la remoción de algún esquema antes de incorporar el nuevo esquema de acción al grafo de planning. En caso que no exista un esquema de acción tal que sus poscondiciones estén presentes en el conjunto de poscondiciones del esquema a insertar, el resultado de la revisión sería el mismo que el de la expansión. Esto se expresa mediante el siguiente postulado.

(**R 3**) **Vacuidad**: Si no existe *B* tal que *B* pertenece a nivel *i* a Π y $B_{Pos} \subseteq A_{Pos}$ entonces $(\Pi \otimes^i A) = (\Pi \oplus^i A)$.

El postulado de vacuidad es una consecuencia de nuestra definición de revisión y el postulado de vacuidad para la operación de contracción de grafos de planning[12].

Los postulados hasta aquí presentados podrían considerarse, meramente, un reflejo de los postulados propuestos por Gärdenfors para la operación de revisión en el contexto de conjuntos de creencias. Sin embargo, es posible vislumbrar propiedades deseables del operador de revisión en el contexto específico de grafos de planning.

Consideremos la noción de completitud introducida en [12]. La idea es asegurar el cumplimiento de las metas, luego de realizar una revisión a nivel *i* por un esquema de acción determinado. Esta propiedad podría expresarse de la siguiente manera. Sea *Met*(Π) el conjunto de metas del grafo de planning Π .

(**R 4**) **Completitud**: *Met*(Π) = *Met* $(\Pi \otimes^i A)$.

El postulado de completitud se debe al modo en que definimos la operación de revisión. Dado que las poscondiciones del esquema de acción a reemplazar son conservadas en el nuevo esquema que se incorpora, es imposible que desaparezcan metas del grafo de planning luego de realizar una revisión por un esquema de acción determinado.

4.2 Construcción

En esta subsección, introducimos una definición constructiva de la operación de revisión de grafos de planning.

Definición 4.5. Sea Π un grafo de planning y sean *A* y *B* esquemas de acción. Definimos la revisión a nivel *i* de Π por *A* como

$$\Pi \otimes^i A = (\Pi \ominus^i B) \oplus^i A), \text{ donde } B_{Pos} \subseteq A_{Pos}$$

De acuerdo a esta definición, con el fin de realizar una revisión por un esquema de acción *A* debemos, en primera instancia, contraer por un esquema de acción cuyas poscondiciones estén incluídas en las poscondiciones del esquema a revisar y, finalmente, realizar una expansión por el esquema de acción *A*.

El siguiente lema resume algunas de las propiedades de interés del operador de revisión.

Lema 4.1. Sea $\otimes^i$ un operador de revisión de acuerdo a la Definición 4.5. Entonces $\otimes^i$ satisface los postulados de *éxito*, *inclusión*, *vacuidad* y *completitud*.

4.3 Ejemplo de Aplicación

A continuación, exhibimos el grafo de planning de la Figura 2 revisado, a nivel 1, por el esquema de acción ({},rotiseria,{cena}). Recordemos que, de acuerdo a nuestra definición de revisión deberíamos, en primer lugar, contraer por ({manosLimpias},cocinar,{cena}) y luego expandir por el esquema ({},rotiseria,{cena}).

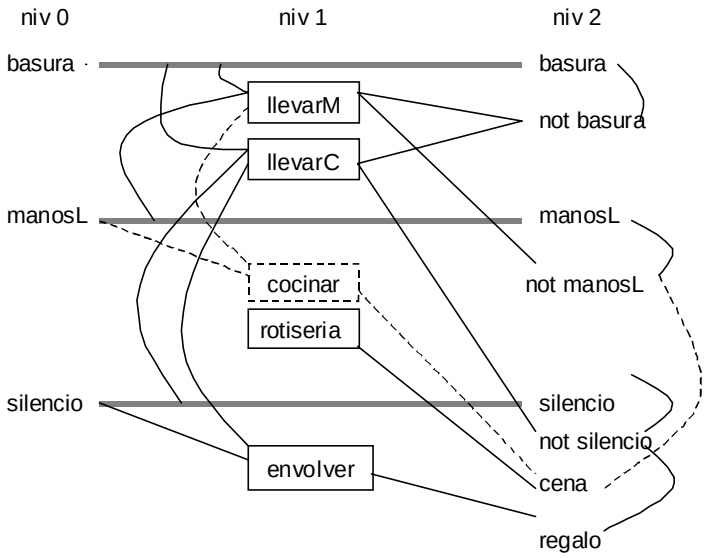

Figura 4

En el gráfico anterior, podemos apreciar (remarcados) los nodos y arcos que desaparecen como resultado de la contracción $\Pi\ \Theta^1$ ({manosLimpias},cocinar,{cena}). Asimismo, se exhibe el cambio en la relación de exclusión mutua provocada por la desaparición de la acción cocinar. Se observa además, la presencia a nivel 1 del esquema de acción ({},rotiseria,{cena}) debido a nuestra definición de revisión. A partir de este nuevo grafo de planning, producto de la revisión del original, Graphplan puede iniciar su segunda fase para intentar extraer la solución al problema.

5 Conclusiones

La contribución principal de este trabajo es la introducción de un modelo para representar revisiones de grafos de planning. Hemos presentado un operador de revisión para grafos de planning y hemos ofrecido un conjunto de postulados para tal operador. Además, la operación de revisión ha sido definida constructivamente.

La definición de este operador de revisión hace posible la reutilización de gran parte del grafo de planning original. En trabajos futuros, se estudiarán definiciones alternativas de operadores de contracción y de revisión, así como también su interrelación.

Referencias

[1] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *J. Artificial Intelligence,* 2(3/4), 1971.

[2] A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of the XIV International Joint Conference of AI*, pages 1636-1642, 1995.

[3] A. Blum and M. Furst. Fast planning through planning graph analysis. *J. Artificial Intelligence*, 90(1-2):281-300, 1997.

[4] Daniel S. Weld. Recent Advances in AI Planning. *AI Magazine*, 1999.

[5] C. Alchourrón, P. Gärdenfors and D. Makinson. On the Logic Of Theory Change: Partial Meet Contraction and Revision Functions. *The Journal of Symbolic Logic,* 50:510-530, 1985.

[6] Peter Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States. The MIT Press, Bradford Books, Cambridge, Massachusetts,* 1988.

[7] M. Falappa. *Teoría de Cambio de Creencias y sus Aplicaciones sobre Estados de Conocimiento*. Tesis Doctoral, Dep. de Cs. de la Computación, Universidad Nacional del Sur, Bahía Blanca, 1999.

[8] Gerardo Parra. *Semi Revisión Plausible en Bases de Creencias*. Tesis de Magister, Dep. de Cs de la Computación, Universidad Nacional del Sur, Bahía Blanca, 1998.

[9] Sven O. Hansson. *A Textbook of Belief Dynamics*. *Kluwer Academic Press*, 1996.

[10] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The Belief-Desire-Intention Model of Agency. In J.P.Müller, M.P.Singh, and A.S. Rao, editors, *Intelligent Agents V* (LNAI Volume 1555), pages 1-10. Springer-Verlag: Berlin, Germany, 1999.

[11] G. Parra, M. Falappa y G. Simari. Replaneamiento en Agentes Inteligentes. Parte I: Expansión. Enviado a *VII International Congress of Information Engineering (ICIE 2001)*. Buenos Aires. Abril de 2001.

[12] G. Parra y G. Simari. *Replaneamiento en Agentes Inteligentes. Contracción de Grafos de Planning*. VII Congreso Argentino de Ciencias de la Computación. Volumen II, páginas 1081-1093. Octubre de 2001 - Universidad Nacional de la Patagonia Austral - El Calafate - Santa Cruz.